# Driveroo

Submitted by: Driverooers

Submitted to: Hao Lac
ICET Department,
School of Engineering Technology and Applied Science
Progress Campus, Block A
Centennial College

Discipline: Software Engineering / Game - Programming

Due Date: 2023-05-23

## Declaration of Sole Authorship

We, Driverooers, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included.

Signed:

- Muhammad Sabeeh, 301184564 (Game - Programming)
- Jin Huang, 301020707 (Game - Programming)
- Tannus Esquerdo, 301288342 (Software Eng Techonology)

Date: 2023-05-23

## Abstract

The proposed project presents Driveroo, a mobile application that seeks to revolutionize the process of scheduling driving lessons by giving consumers a practical and tailored experience. Users of the app may search for driving instructors nearby, look at teacher profiles and reviews, set up appointments, and conveniently book lessons using the app. It responds to the rising desire for a quicker, easier way to locate driving instructors, particularly in regions with high demand.

The importance of Driveroo lies in its capacity to simplify the booking procedure, increase accessibility for students, and broaden the visibility of driving instructors. The software streamlines the scheduling and payment procedures, making it simpler for users to book courses. It also integrates payment channels. Additionally, teacher discussions guarantee a unique learning path for every user.

Potential users should download the Driveroo app and use it to look for driving instructors, make appointments for consultations, and reserve lessons in order to take full advantage of its features. Users may locate and book driving instructors in a more effective and easy manner by making use of the app's features and functions. The number of registered driving instructors, customer contentment, and favourable app store reviews will be used to gauge the project's success since they show the app's influence and worth within the driving instruction sector.

## Table of Contents

## List of Figures

## List of Tables

## 1.0 INTRODUCTION

The technological issue addressed in this project is the slowness and absence of a simplified approach to booking driving lessons. The current techniques for locating and hiring driving instructors sometimes include time-consuming searches, restricted access to teacher information, and manual scheduling. The goal of this project is to create a mobile application called Driveroo to address this issue and provide a more effective way for learners to identify and book driving teachers.

This Technical Report (TR) describes the work done to propose and detail the development of the Driveroo app. The objective is to provide a user-friendly platform that streamlines the process of identifying and hiring driving instructors, enhancing the entire learning experience for students.

The report includes sections such as Introduction, Methodology and Results, Conclusions, Recommendations, and Appendices. It covers a literature review, a proposed solution, and user role modeling. The report focuses on planning, analysis, and user-centric aspects of the project, omitting specific implementation details. Procedures used include brainstorming, group work, user role modeling, release and iteration planning, progress monitoring, and acceptance testing. The scope encompasses the development of the Driveroo app and the processes involved in user role modeling, release planning, and iteration planning. It provides a comprehensive understanding of project objectives, user requirements, and the planning and testing processes involved in app development.

The primary goal is to create and introduce the Driveroo mobile application, which will let users look up driving instructors, review their profiles and reviews, schedule consultations, and book lessons all through the app. The emphasis on personalized experiences through talks with teachers prior to personalized scheduling of classes is what makes this concept distinctive.

## 2.0 METHODOLOGY AND RESULTS

### 2.1 Literature Review

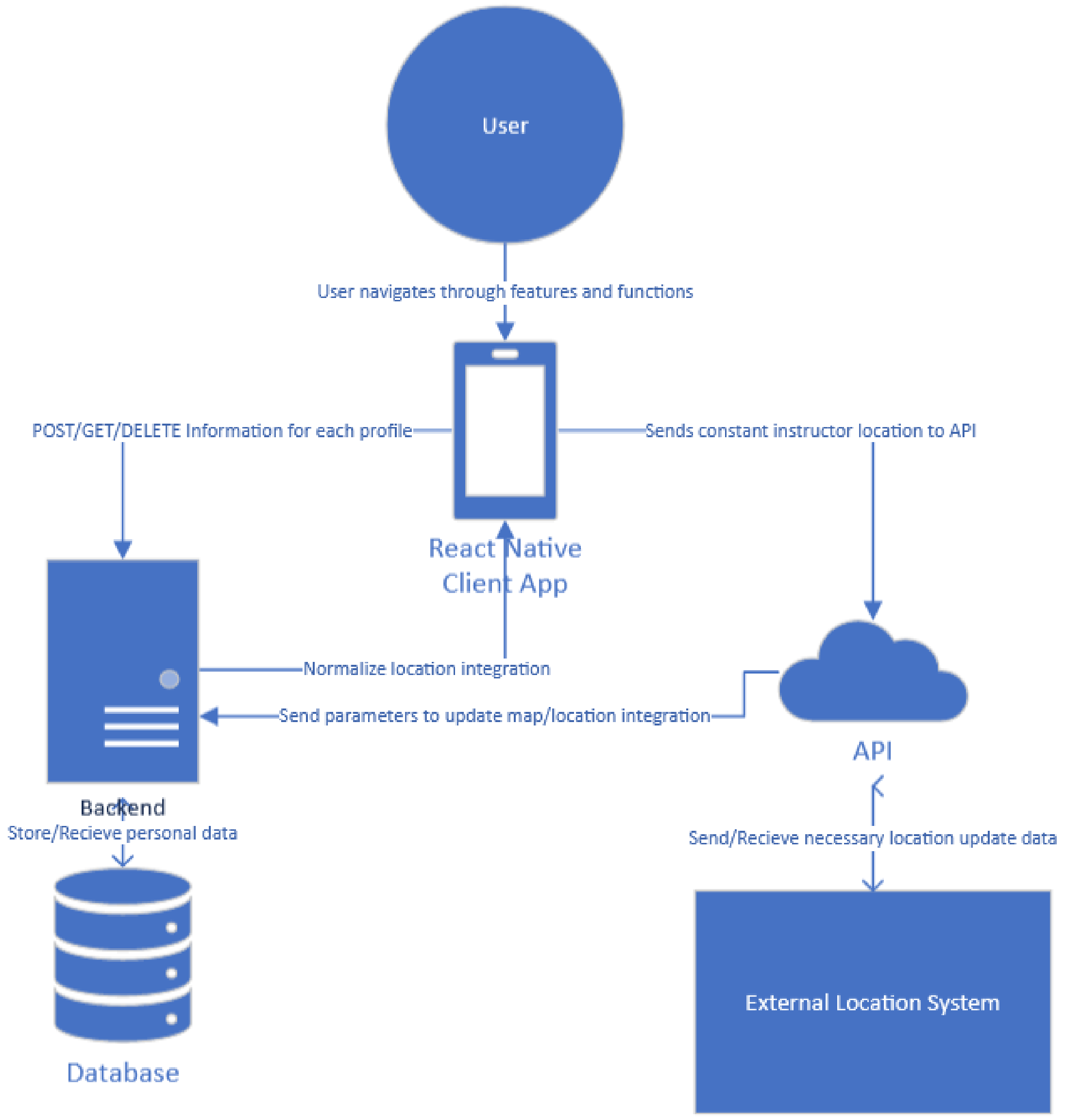
Existing methods for identifying and booking driving instructors have shortcomings that Driveroo can overcome. These solutions are often tied to specific firms or driving schools, limiting independent teachers' access to new students. As a result, individual instructors have few possibilities to extend their student base beyond word-of-mouth recommendations and self-promotion. Furthermore, the existing process is inefficient and inconvenient. Users are frequently faced with a time-consuming process of picking appropriate packages, submitting required information, and being given random instructors without the chance to engage or discuss with them beforehand. Furthermore, scheduling is rigid and inflexible.

Despite these flaws, existing methods have some advantages, such as verification and recruitment processes. Driving school instructors are thoroughly vetted to ensure they have the required qualifications and credentials. Schools frequently hire instructors straight from graduating classes of teaching programs, resulting in a greater pool of trained teachers. These verification techniques instill trust and professionalism in users by providing them with confidence in the teachers' abilities.

## 2.2 Proposed Solution

Our system's strength to greatly ease the process of finding driving instructors for both consumers and employees is one of its main advantages. Users may easily access a wide network of driving instructors in their area by using our application, which is based on their general location. Users benefit from increased convenience and time and effort savings because of this functionality. Additionally, our platform provides a great level of flexibility, enabling customers to communicate openly with teachers about session scheduling and specific needs. The platform includes a strong profile matching system as well, allowing for easy pairing based on things like particular needs, instructor ratings, financial concerns, and schedule compatibility. This capability helps create a highly effective and customised experience, which ultimately optimises the user journey.

There are a few flaws in our system that need to be considered. First, it depends on the user's area having a sufficient number of driving instructors. If there aren't enough instructors, consumers may not be able to book courses through the app**.** Furthermore, the lengthy verification process for driving instructors that is required by regulatory rules. Although this procedure ensures compliance, it could cause delays when adding new teachers and growing the instructor network for the app. To provide a clear picture of the system's limitations, it is critical to identify and address these deficiencies.


User
User navigates through features and functions
POST/GET/DELETE Information for each profile
Sends constant instructor location to API
React Native Client App
Normalize location integration
Send parameters to update map/location integration
API
Backend
Store/Recieve personal data
Send/Recieve necessary location update data
Database
External Location System

https://centennialcollegeedu-my.sharepoint.com/:u:/g/personal/jhuan145_my_centennialcollege_ca/EY4Pg4T08kxLq2_WFovnx3QBGQ2Qwz5mZ5ew3zRpoGFl8g?e=N6H3B0&action=edit&or=PrevEdit

## 2.3 User Role Modelling

### 2.3.1 Brainstorm and Group (NOT PLURAL)

Show the results of your brainstorming session for identifying initial user roles and how they are organized (see Figure 1). Discuss each user role identified and the arrangement of Figure 1.

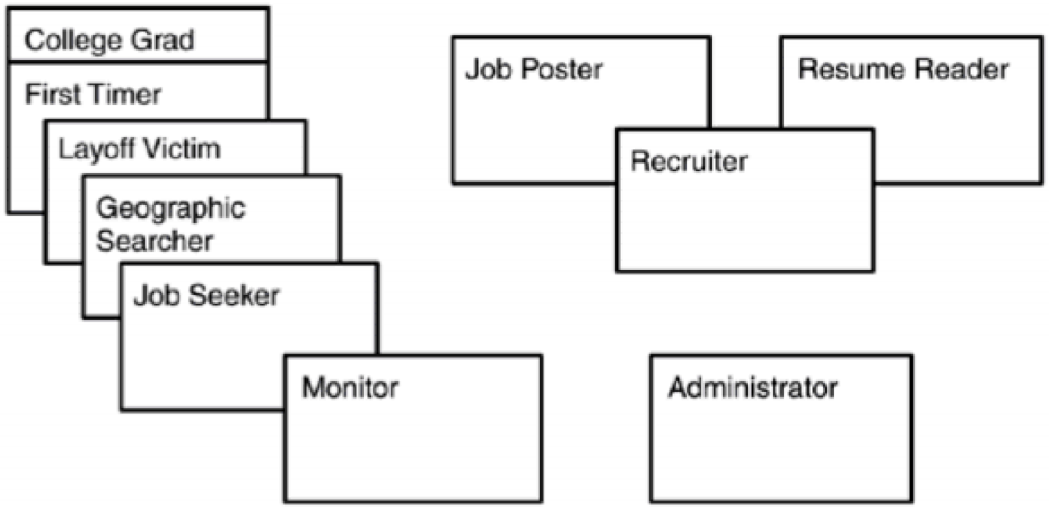


Figure 1: Organizing the user role cards on a table [1].

## 2.3.2 Consolidated User Roles

Show the consolidated user roles (see Figure 2). Discuss the results of Figure 2, focusing on why some roles were merged, removed, and/or added.

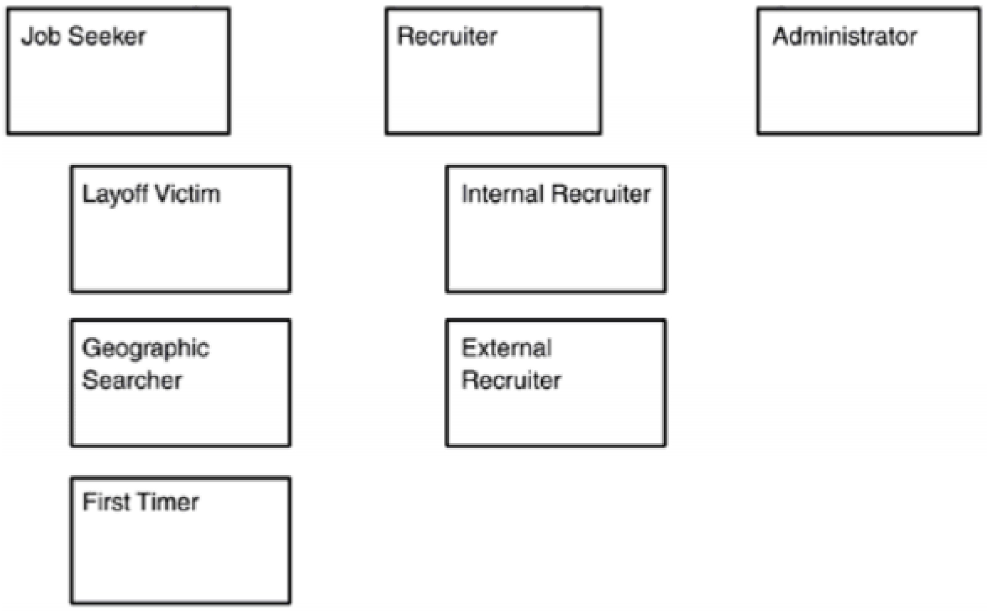


Figure 2: The consolidated role cards [1].

### 2.3.3 Description of User Roles and Persona

For each consolidated role from the above section 2.3.2, include detail that answer at least the following questions:

- The frequency with which the user will use the software.
- The user's level of expertise with the domain.
- The user's general level of proficiency with computers and software.
- The user's level of proficiency with the software being developed.
- The user's general goal for using the software. Some users are after convenience, others favour a rich experience, and so on.

Include personas here (optional).

### 2.3.4 Additional Documentation

For this section, include the video(s) from your workshop showing how your team:

1. Brainstormed for the initial set of user roles.
2. Organized the initial set of roles.
3. Consolidated and condensed the roles.
4. Generated detailed description of each consolidated role.

Provide the file name and URL to the video(s) in your shared folder or YouTube channel.

## 2.4 Release 1.0

### 2.4.1 User Stories

The following are required for this section:

1. Show and discuss the results of your low-fidelity prototype generated during your story-writing workshop (a sample of a “consolidated” low-fidelity prototype is illustrated in Figure 3).
2. Provide your definition of story point.
3. Show the stories created during the story-writing workshop. You can submit scanned images of your index cards (both front and back). Figures 4 to 7 illustrates a single story with variation on the *Note*s (Figures 4 and 5), acceptance tests shown on the back of the index card (Figure 6), and a constraint, or non-functional requirement (Figure 7).
4. Prioritized stories based on the MoSCoW rule as illustrated in Tables 1 and 2 (see also *User Stories* deliverable).

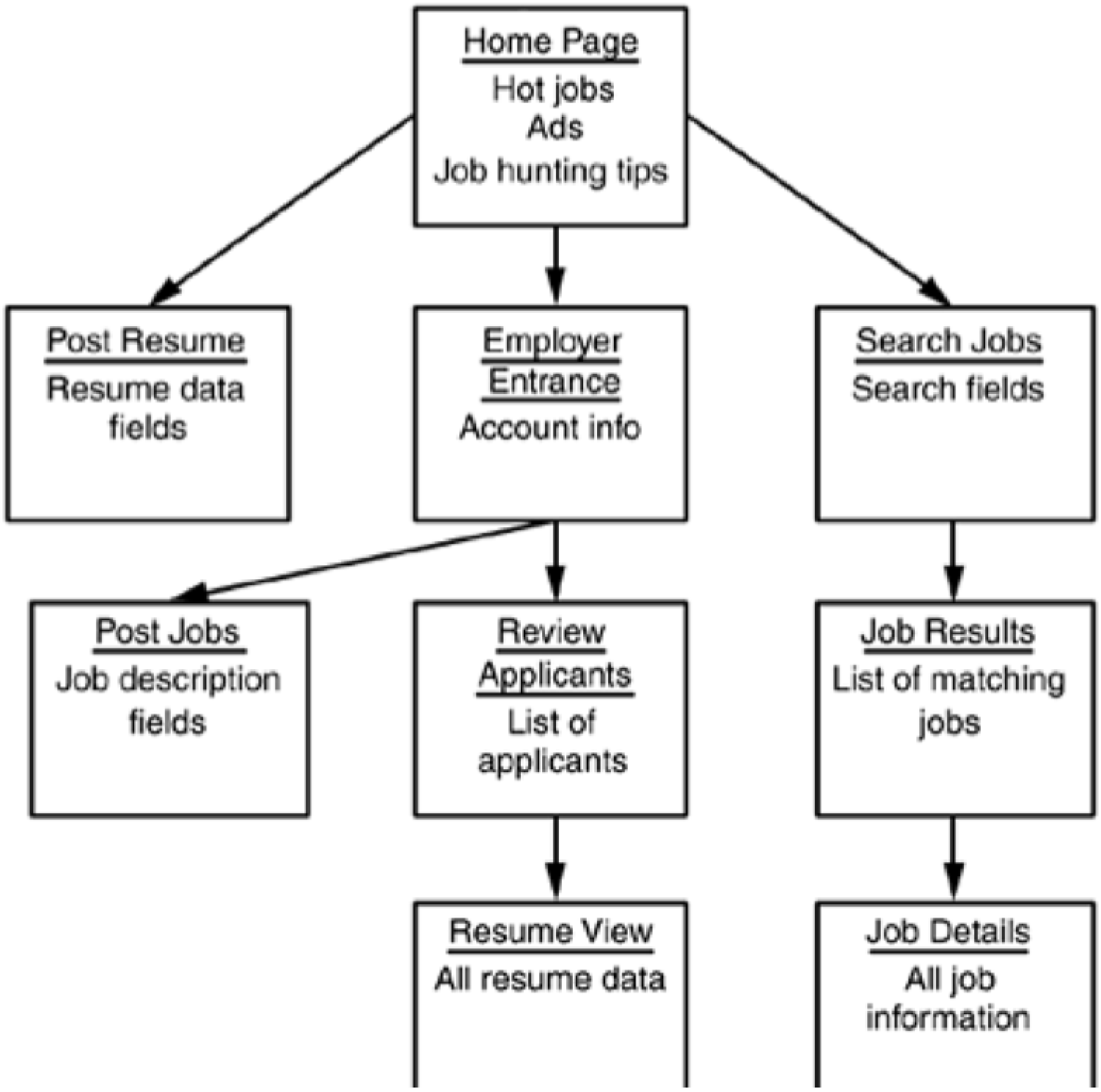
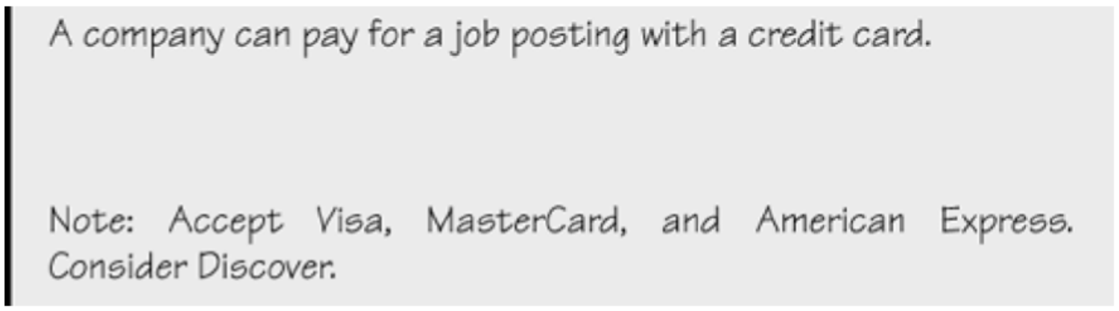
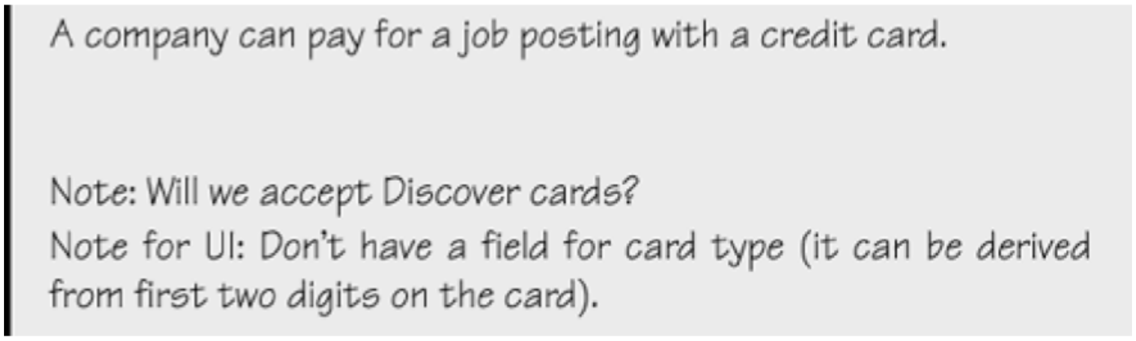


Figure 3: Example of a “consolidated” low-fidelity prototype. Note that each “individual” low-fidelity prototype is developed for each user role [1].


A company can pay for a job posting with a credit card.

Note: Accept Visa, MasterCard, and American Express. Consider Discover.

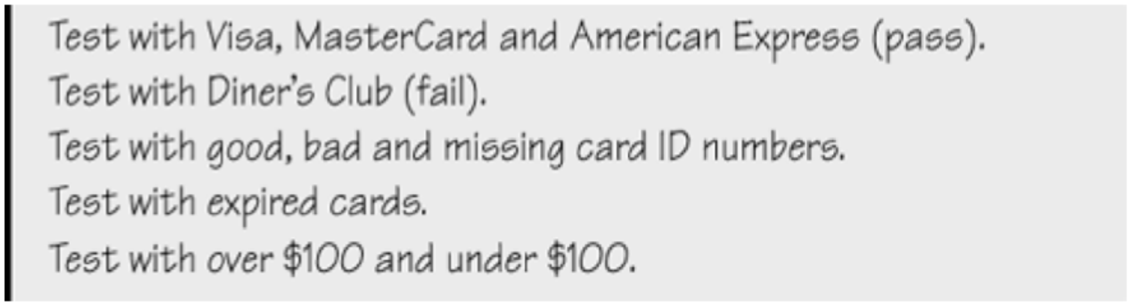
Figure 4: A story with notes providing additional detail [1].


A company can pay for a job posting with a credit card.

Note: Will we accept Discover cards?

Note for UI: Don't have a field for card type (it can be derived from first two digits on the card).

Figure 5: The revised front of a story card with only the story and questions to be discussed [1].


Test with Visa, MasterCard and American Express (pass).

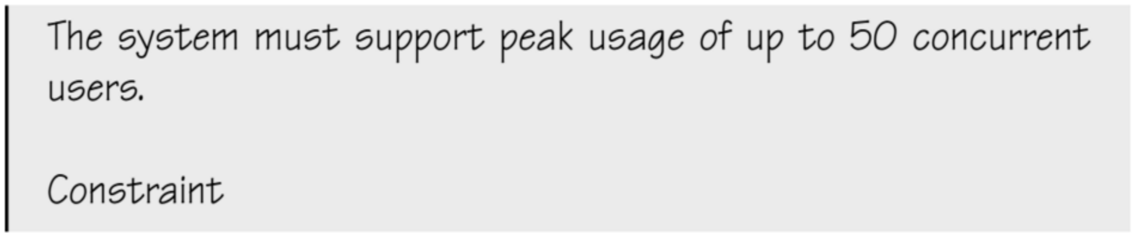
Test with Diner's Club (fail).

Test with good, bad and missing card ID numbers.

Test with expired cards.

Test with over $100 and under $100.

Figure 6: Details that imply test cases are separated from the story itself. Here they are shown on the back of the story card [1].


The system must support peak usage of up to 50 concurrent users.

Constraint

Figure 7: An example of a constraint story card [1].

Figure 8 illustrates a possible electronic representation of a physical story card. The left column represents the front of the card while the right column represents the back of the card.

| | |
|---|---|
| A Company can pay for a job posting with a credit card.<br><br>Note: Will we accept Discover cards?<br>Note for UI: Don't have a field for card type (it can be derived from the first two digits on the card)<br><br>Estimate: 3 hrs. | Test with Visa, MasterCard and American Express.<br>Expected outcome: the system should automatically display a label of the card type.<br><br>Test with Diner's Club.<br>Expected outcome: the system should prompt the user for a Visa, MasterCard or American Express card.<br><br>...*<rest of the Tests follows>* |

Figure 8: Possible electronic representation of a physical story card.

Table 1: The Must-Have stories for Release x.y [1].

| Story | Estimate |
|---|---|
| A user can do a basic simple search that searches for a word or phrase in both the author and title fields. | 1 |
| A user can put books into a "shopping cart" and buy them when she is done shopping. | 1 |
| A user can remove books from her cart before completing an order. | ½ |
| To buy a book the user enters her billing address, the shipping address and credit card information. | 2 |
| A user can establish an account that remembers shipping and billing information. | 2 |
| Orders made on the website have to end up in the same order database as telephone orders. | 0 |
| An administrator can add new books to the site. | 1 |
| An administrator can delete a book. | ½ |
| An administrator can edit the information about an existing book. | 1 |
| The system must support peak usage of up to 50 concurrent users. | 0 |

Table 2: The Should-Have stories for Release x.y [1].

| Story | Estimate |
|---|---|
| A user can search for books by entering values in any combination of author, title and ISBN. | 1 |
| A user can edit the credit card information stored in her account. | ½ |
| A user can edit the shipping and billing addresses stored in her account. | 1 |
| A user can see what books we recommend on a variety of topics. | 4 |

### 2.4.2 Additional Documentation

For this section, include the video(s) from your workshop showing how your team:

1. Brainstormed for stories and generated the low-fidelity prototype (story writing workshop).
2. Estimated stories using the Wideband Delphi approach.
3. Prioritized stories using the MoSCoW rule.

Provide the file name and URL to the video(s) in your shared folder or YouTube channel.

### 2.4.3 Release Plan 1.0

The following are required for this section[1]:

1. Provide the product development roadmap.
2. Provide the iteration length and the release date.
3. The refine priorities of the Must- and Should-Have stories by organizing the stories into groups that have a high likelihood of being performed together.
4. The actual release plan.
5. Place the contents of your paper prototype in Appendix A (Design Document).

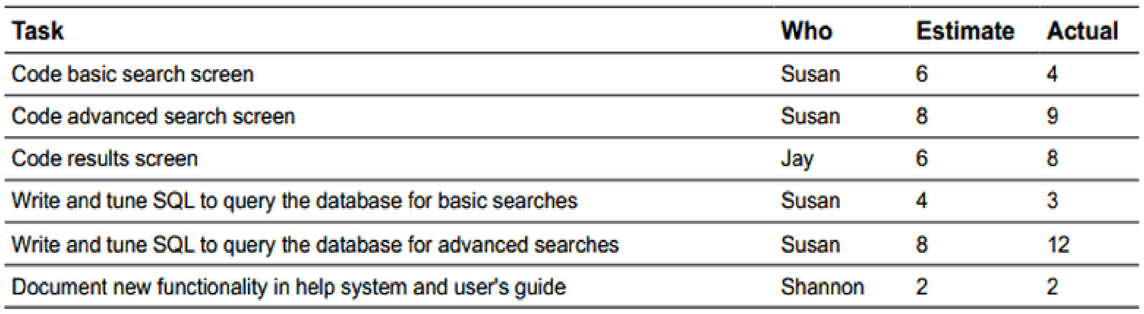
[1] See *The Release Plan* deliverable.

### 2.4.4 Iteration Plan (Release 1.0)

The following are required for this section:

1. Present each iteration plan with tables showing disaggregated tasks per story; a sample is shown in Table 3. See also the *Planning an Iteration* deliverable.
2. Discuss any discrepancies between the estimated and actual ideal time required to complete the tasks for the Table mentioned above.

Table 3: Disaggregated tasks per story [1].



| Task | Who | Estimate | Actual |
|---|---|---|---|
| Code basic search screen | Susan | 6 | 4 |
| Code advanced search screen | Susan | 8 | 9 |
| Code results screen | Jay | 6 | 8 |
| Write and tune SQL to query the database for basic searches | Susan | 4 | 3 |
| Write and tune SQL to query the database for advanced searches | Susan | 8 | 12 |
| Document new functionality in help system and user's guide | Shannon | 2 | 2 |

### 2.4.5 Additional Documentation

For this section, include 1 of 4 videos from your Iteration Planning meetings (recall that you have a total of 4 Iteration Planning meetings)[2]:

1. Showing how your team disaggregated stories into their constituent tasks.
2. How developers on your team volunteer and take responsibilities for tasks.

Provide the file name and URL to the video(s) in your shared folder or YouTube channel.

[2] Indicate which iteration the video corresponds to. If you decide to submit a video in Release 1.0, then you do not need to include an *Additional Documentation* section for Release 2.0.

### 2.4.7 Acceptance Tests for Release 1.0

The following are required for this section:

1. A table of stories and their associated acceptance tests for this Release as shown below in the sample in Table 5.
2. The link to your video demo for Release 1.0 stored either in a cloud drive, or your YouTube channel.

Table 4: Stories, acceptance tests, and contributors for Release 1.0 (Green=Passed; Red=Failed).

| **Full description of user story** | **Acceptance test(s)** | **Name(s) of contributing Developer(s)** |
|---|---|---|
| As an User, I can … so that ….[3] | Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ... | Susan Smith,<br>Jay Johnson |
| As an Administrator, I can … so that ….[4] | Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ... | Susan Smith,<br>Jay Johnson,<br>Shannon Shore,<br>George Gavinson |
| As an User, I can … so that …. | Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ... | Jay Johnson,<br>Shannon Shore,<br>George Gavinson |
| As an User, I can … so that ….[5] | Test with inputs ….<br>Expected outcome: ... | Shannon Shore |
| As a Guest, I can … so that …. | Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ...<br><br>Test with inputs ….<br>Expected outcome: ... | Susan Smith,<br>Jay Johnson,<br>Shannon Shore,<br>George Gavinson,<br>Abbey Appleby,<br>Brian Bolt |

[3] Green colour code indicates that all tests passed successfully as intended.
[4] Red colour code indicates that at least one test unintendedly failed.
[5] When all tests for a given story fails, this may suggest that implementation of the story has not even begun and indicates poor planning on the part of the team.

*<Insert url to video demo of Release 1.0 here>*

## 2.5 Release 2.0

Release 2.0 has essentially the same structure as Release 1.0.

### 2.5.1 User Stories

If your team wrote enough stories to cover up to or beyond Release 2.0 during your first story-writing workshop as described in the *User Stories* section 2.4.1, then your team will not need to hold a second formal workshop.

If a second workshop was held, submission for this section is the same as section 2.4.1.

### 2.5.2 Additional Documentation

Include this section in your Technical Report only if your team required a second formal story-writing workshop. If a second workshop was held, submission for this section is the same as section 2.4.2.

### 2.5.3 Release Plan 2.0

The requirements for this section are the same as section 2.4.3; update or add sections if required.

### 2.5.4 Iteration Plan (Release 2.0)

The requirements for this section are the same as section 2.4.4.

### 2.5.5 Additional Documentation

This section is required ONLY IF your team submitted materials for section 2.4.5.

### 2.5.7 Acceptance Tests for Release 2.0

The requirements for this section follow the same requirements as in section 2.4.7 except acceptance testing is for stories allocated for Release 2.0 and incomplete stories subsequently moved from Release 1.0.

## 3.0 CONCLUSIONS

A conclusion interprets the data found in the Body. It is reasoned judgment and not opinions. Consider the variables. Relate cause and effect. Analyze, evaluate, make comparisons and contrasts. Base the conclusion on fact.

## 4.0 RECOMMENDATIONS

Recommendations are not required for all studies. They suggest a course of action and would generally be provided when there are additional areas for study, or if the reason for the TR was to determine the best action going forward.

## CREDITS, LICENSE, AND REFERENCES

### Credits

Provide any credits here. The following are examples:

Author of the template graphic layout : Hao Lac

<haolac.at.centennial@gmail.com>

Author of the template explanation text : John Doe

<john.doe@centennialcollege.ca>

### License

State the license granted with your system. For example:

### References

[1] Cohn, Mike. 2004. *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional.

## APPENDIX A (DESIGN DOCUMENT)

Traditional approaches to software development, in contrast to that of Agile approaches, place a great deal of emphasis on upfront design. The Agile approach to design is quick sessions that seek the simplest solution and then incrementally build on that solution. A quick design session can include the use of CRC cards that can ultimately lead to the generation of UML diagrams.

Using Agile approaches to software development does not mean you are limited to using only Agile techniques. If you feel that a technique (e.g., use case or interaction design scenario) is more suitable, or better conveys the features of your system to your users, then use it.

In this section, you are required to submit and discuss the following:

- A paper prototype of your application/system.
- Any design work your team has done in developing your system including CRC cards, UML diagrams, ERD diagrams, use cases, interaction design scenario, etc.

# APPENDIX B (TEST PLAN)

## 1.0 Introduction

### 1.0.1 Goals

Summarize the testing goals for the project.

### 1.0.2 Assumptions

Any assumptions which may affect the understanding or execution of this plan should be recorded here.

### 1.0.3 Risks And Assets

Describe the elements (software or hardware) that are not part of your application but still may impact its correctness and must be checked.

Describe the elements that might positively influence testing on the project.

## 2.0 Scope

### 2.0.1 Features To Be Tested

Describe the features and functions that will be tested during the project. This should include functional and non-functional requirements.

### 2.0.2 Features Not To Be Tested

Describe the features that will not be tested and reason why.

## 3.0 Testing Procedures

Describe the testing procedures that the project will use. This includes the test lifecycle, types of testing, test objectives, and test criteria.

### 3.0.1 Test Objectives

Describe the objectives of the testing process.

### 3.0.2 Types Of Testing

Describe the types of testing that the project will use.

#### 3.0.2.1 Unit Testing

Describe the strategy for unit testing of the individual subsystems. This includes an indication of the subsystems that will undergo unit tests or the criteria to be used to select subsystems for unit test. Test cases are NOT included here.

#### 3.0.2.2 Integration Testing

Specify the integration testing strategy used. Describe the tests that will be performed in order to verify the interfaces between the subsystems of the software system. This section includes a discussion of the order of integration of subsystems. Test cases are NOT included here.

#### 3.0.2.3 Acceptance Testing

Specify the strategy for testing the software once it has been deployed. This section includes a discussion of the order of acceptance by software function. Test cases are NOT included here.

#### 3.0.2.4 Stress Testing

Identify the limits under which the program is expected to perform (memory constraints, disk space constraints, etc).

#### 3.0.2.5 Performance Testing

Refer to the functional requirements that specify acceptable performance.

### 3.0.3 Testing Tools

Describe the tools that you will use for testing.

## 4.0 Schedule and Deliverables

Describe the test deliverables that will be created during the project lifecycle. Include two tables, one for the schedule of tasks, another for the list of deliverables:

- Acceptance test
- Unit test
- System/Integration test
- Stress test
- Performance test
- Screen prototypes
- Defect reports and summaries
- Test logs and reports

Describe the reports that will be generated by the testing process.

Examples include:

Test Summary Report - A final report of the testing results from the project. Can include items such as total number of test cases, number of test cases executed, % test cases passed, etc.

## APPENDIX C (END-USER & ADMINISTRATOR MANUALS)

In this section, include a user manual for your system/application. The user manual should include the following items:

1. Instructions on how to install and configure your system/application, documenting all external software dependencies that need to be setup manually.
2. A user guide for the administrator (use screen shots of your system/application and briefly discuss each screen shot).
3. A user guide for the normal user (use screen shots of your system/application and briefly discuss each screen shot).

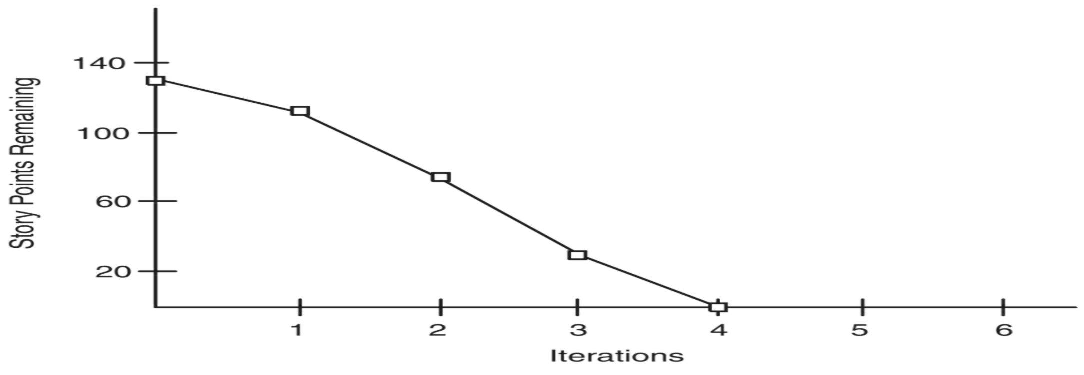
## APPENDIX D (PROGRESS MONITORING)

Your team is required to report two items related to progress monitoring in this appendix. The first item is a table summarizing progress and changes during a release with supporting discussion; a sample is shown in Table 5. Notice in Table 5 that all iterations are shown per Release[6]. Also, see *Table 1* in the *Measuring and Monitoring Progress* deliverable.

Table 5: Progress and changes for all iterations [1].

| | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 |
|---|---|---|---|---|
| Story points at start of iteration | 130 | 113 | 78 | 31 |
| Completed during iteration | 45 | 47 | 48 | 31 |
| Changed estimates | 10 | 4 | –3 | |
| Story points from new stories | 18 | 8 | 4 | |
| **Story points at end of iteration** | **113** | **78** | **31** | **0** |

The second item is an iteration burndown chart (see Figure 9) reflecting the data from Table 5.



[6] For subsequent Releases, do NOT restart numbering the Iteration. For example, let us assume that we have another Release (i.e., Release 2.0), we would continue numbering our Iterations as *Iteration 5, Iteration 6,* and so on.

Figure 9: Iteration burndown chart for data from Table 5.